PARTITIONING DATA FOR ERROR CORRECTION

FIELD OF THE INVENTION

[0001] The present invention relates to error correction and, more particularly, to partitioning data for error correction.

BACKGROUND

[0002] Error codes are commonly used in electronic systems to detect and/or correct data errors, such as transmission errors or storage errors. One common use of error codes is to detect and correct errors with data stored in a memory of a computer system. For example, error correction bits, or check bits can be generated for data prior to storing data to one or more memory devices. The error or correction bits are appended to the data to provide a data structure that is stored in memory. When the data is read from the one or more memory devices, the check bits can be used to detect or correct errors within the data. Errors can be introduced, for example, either due to faulty components or noise in the computer system. Faulty components can include faulty memory devices or faulty data paths between the devices within the computer system, such as faulty pins.

[0003] Error management techniques have been developed to mitigate the effects associated with these errors. One simple technique used for personal computers is known as parity checking. Parity checking utilizes a single bit associated with a piece of data to determine whether there is a single bit error in the data. Parity checking cannot detect multiple bit errors and provided no means for correcting errors. A more sophisticated system, such as a server, uses error correction codes (ECCs) to detect and correct some errors. An error correction code (ECC) consists of a group of bits, or codes, associated with a piece of data. A typical ECC system may use eight ECC bits (check bits, correction bits) for a 64-bit piece of data. The ECC bits provide enough information for an ECC algorithm to detect and correct a single bit error, or to detect double bit errors.

[0004] One error correction feature employed by servers is referred to in the industry as chipkill. The term chipkill refers to the ability to correct multiple bit errors in memory, where multiple bit errors are based on the width of the memory device. For example, for a 32Mbit dynamic random access memory (DRAM) device that is 4 bits wide, a system that supports a chipkill function would be able to correct a

4-bit wide error in the memory device. Thus, the failure of an entire DRAM chip during a DRAM cycle (e.g., read operation, write operation) organized into a 4-bit width configuration that supports chipkill would not cause the system to fail. Chipkill allows a system to operate in the event of multiple bit errors in any one memory device.

SUMMARY

[0005] The following presents a simplified summary of the invention in order to provide a basic understanding of some aspects of the invention. This summary is not an extensive overview of the invention. It is intended to neither identify key or critical elements of the invention nor delineate the scope of the invention. Its sole purpose is to present some general concepts of the invention in a simplified form as a prelude to the more detailed description that is presented later.

[0006] In one embodiment, the invention encompasses a method of writing to a plurality of memory devices of a memory system. A code word to be stored in the memory system is received and partitioned into a plurality of nibbles. The nibbles have a bit width that corresponds to widths of the plurality of memory devices. The partitioned code word is stored into the plurality of memory devices by storing a plurality of successive nibbles of the block of data into each of the plurality of memory devices.

[0007] In another embodiment, the invention encompasses a method of reading from a plurality of memory devices of a memory system. A plurality of chunks of data are read from the plurality of memory devices where each chunk comprises a nibble from each of the plurality of memory devices and the nibbles have a width corresponding to a width of one or more corresponding memory devices. The nibbles from the plurality of chunks are combined to generate a code word where the nibbles from each of the plurality of memory devices are adjacent in the code word.

[0008] In yet another embodiment, the invention encompasses a memory system. A data buffer reads and writes chunks of bits from a plurality of memory devices. A data combiner receives the chunks of bits read by the buffer and forms a code word by separating the chunks into a plurality of nibbles having widths corresponding to at least one of the memory devices and arranging the nibbles having the same relative position in their respective chunks adjacent to each other in the code word. A data separator receives a code word and separates the code word into a plurality of chunks for storage by the buffer into the memory devices. The data

separator partitions the code word into a plurality of sequential nibbles with widths corresponding to the width of at least one of the plurality of memory devices, partitions the nibbles into groups of adjacent nibbles, and generates the chunks of bits having only one nibble from each group of nibbles with the nibbles of each group in the same relative position in their respective chunks.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] For the purpose of illustrating the invention, there is shown in the drawings a form that is presently preferred; it being understood, however, that this invention is not limited to the precise arrangements and instrumentalities shown.

[0010] Figure 1 is a flow chart illustrating a method for writing a data block into a memory system according to an embodiment of the invention;

[0011] Figure 2 is a partial block diagram of a memory system according to an embodiment of the invention;

[0012] Figures 3 is a block diagram illustrating storage of nibbles in a memory system according to an embodiment of the invention;

[0013] Figure 4 is a is flow chart illustrating a method for reading a data block from a memory system according to an embodiment of the invention; and

[0014] Figures 5 is a block diagram illustrating storage of nibbles in a memory system according to embodiments of the invention.

DETAILED DESCRIPTION

[0015] The present invention relates generally to systems and methods for detecting and correcting errors in code words. A code word comprising a plurality of error detection and correction bits and data bits is partitioned into nibbles. Nibbles adjacent to each other in the code word are transmitted and/or stored on a common physical transmission or storage medium. In the case of a failure of the common physical medium, when the code word is reconstructed from the physical medium, the failure causes errors in adjacent nibbles. In other words, the bits in the code word that are erroneous due to the failure of the common physical medium form a contiguous string in the code word.

[0016] An error correction code may be configured to correct more erroneous bits if the bits are adjacent to each other rather than if they are dispersed (not adjacent) through a code word. By partitioning the data according to an embodiment of the invention, is may be possible to correct the erroneous bits resulting

from the failure of the common physical medium with less overhead of error detection and correction bits.

[0017] Referring to the drawings, in which like reference numerals indicate like elements, a method of writing to a plurality of memory devices according to an embodiment of the invention is described with reference to the flow chart 100 of Figure 1 and the memory system 200 shown in Figure 2.

[0018] The error detection and correction module 210 receives a data block in step 102. The error detection and correction module 210 generates error detection and correction bits from the data block in step 104 and combines the data block and the error detection and correction bits to form a code word in step 106. The data separator 208 receives the code word and partitions it into nibbles having bit widths corresponding to the bits widths of the memory devices 202 and partitions the nibbles into groups of adjacent nibbles in step 108. The number of nibbles in a group equals the bit width of the code word divided by the bit width of an address line across the memory devices.

[0019] The data separator 208 generates chunks of nibbles having one nibble from each group of nibbles in step 110. The buffer 204 receives the chunks of nibbles and stores them in the memory devices 202 in step 112 so the nibbles in each group are stored in the same memory device 202.

[0020] The method described above with regard to Figure 1 is illustrated in an example with reference to the code word and memory devices shown in Figure 3. The code word shown in Figure 3 has twelve nibbles A-L and there are three memory devices 0-2 where each memory device has a bit width equal to the bit width of a nibble. The nibbles are formed into groups of four adjacent nibbles per group as illustrated in Figure 3. Four chunks labeled 0-3 are formed where each chunk has one nibble from each group and is written to an address line in the memory devices 202 as illustrated in Figure 3.

[0021] By partitioning the code word as illustrated in Figure 3, if memory device 1 fails, the resulting code word read from the memory devices will be [A,B,C,D,X,X,X,X,I,J,K,L] where the X's designate erroneous nibbles (i.e., nibbles having errors due to the failure of memory device 1). The erroneous nibbles are adjacent to each other in the reconstructed code word and may therefore be correctable with an error code that uses less correction bits than might be necessary to correct four erroneous nibbles that are not adjacent.

[0022] A method of reading a code word from a plurality of memory devices according to an embodiment of the invention is described with reference to the flow chart 400 of Figure 4 and the memory system 200 shown in Figure 2. The buffer 204 reads a plurality of chunks from the memory devices 202 in step 402 where each chunk includes a nibble from each of the plurality of memory devices 202, the nibble having a bit width corresponding to the width of the corresponding memory device 202.

[0023] The data combiner 206 receives the plurality of chunks from the buffer 204 and generates a code word from the nibbles in the plurality of chunks in step 404. The nibbles are positioned in an order in the code word so the nibbles from each of the plurality of memory devices are adjacent in the code word. The error detection and correction module 210 receives the code word and performs an error detection and correction algorithm on the code word to detect and/or correct errors in the code word. The error detection and correction module 210 then extracts the data bits from the code word and outputs a corresponding data block.

[0024] With reference to the exemplary code word and memory devices of Figure 3, a code word is read from the memory devices as follows. In this example, a code word is formed from four chunks (labeled 0-3) of nibbles. The buffer 204 reads the four chunks from the memory devices where the chunks and their corresponding nibbles are identified as follows.

Chunk	Nibbles
Chunk 0	[A,E,I]
Chunk 1	[B,F,J]
Chunk 2	[C,G,K]
Chunk 3	[D,H,L]

[0025] The data combiner 206 receives the four chunks of nibbles from the buffer 204 and reconstructs their nibbles into order from A-L to form the code word. The code word is then further processed by the error detection and correction module 210 in steps 406 and 408 as described above.

[0026] The examples illustrated with regard to Figure 3 are for illustrative purposes only and the invention is generally applicable to other configurations of code words, nibble, chunks, and memory devices. Another example is illustrated with reference to Figure 5 where a code word has a number CW of bits and the memory system includes a number K of memory devices each having a number W of bits in width. An address line across the plurality of K memory devices

includes a number MD of bits, in this case MD=K*W bits, because the K memory devices all have the same bit width W. The number CW of bits in the code word is a multiple of the size MD of the address line.

[0027] A code word to be written to the memory devices is partitioned by the data separator 208 into a number N of nibbles each having W bits where N=(CW/W) and the nibbles are sequentially numbered from 0 to N-1. The nibbles are partitioned into a number M of chunks where M=(CW/MD) and each chunk includes K nibbles defined as follows:

Chunk[C] = {nibble[C+(K-1)*M], nibble[C+(K-2)*M], ...
... , nibble[C+(1)M], nibble[C+(0)*M]}

for C=0, 1, ...M-1.

[0028] A code word to be read from memory is formed by reading M=CW/MD chunks from the K memory devices where each chunk comprises one nibble having a width W from each of the memory devices. The data combiner 206 arranges the nibbles to form a code word where the nibbles from each memory device are adjacent as follows:

Code Word = [nibble(M-1,K-1), nibble (M-2,K-1), ..., nibble(0,K-1),
nibble(M-1,K-2), nibble (M-2,K-2), ..., nibble(0,K-2),
...
nibble(M-1,0), nibble (M-2,0), ..., nibble(0,0)]

where each nibble is designated as nibble(x,y) where x is its chunk identifier and y is its sequential position within the chunk.

[0029] In an embodiment, the system 200 includes 36 memory devices 202 with a width W of 4 bits each and the code word has a width CW of 288 bits. When writing to the memory devices 202, the data separator 208 partitions the code word into 72 sequentially numbered nibbles of 4 bits. The data separator 208 partitions the nibbles into two nibbles, one with even-numbered nibbles and another with odd-numbered nibbles. The buffer 204 stores the first and second chunks in respective address lines of the 36 memory devices so each pair of adjacent nibbles from the code word is stored in a different one of the 36 memory devices.

[0030] When reading from the 36 memory devices, the buffer 204 reads two chunks of 144 bits each from the memory devices. The first chunk has 36 4-bit "first" nibbles, one from each of the memory devices. The second chunk has 36

4-bit "second" nibbles, one from each of the memory devices. The data combiner 206 combines the nibbles from the first and second chunks to generate a code word having the first and second nibbles from each memory device adjacent to each other. The error detection and correction module 210 receives the code word from the data combiner 206 and performs an error detection and correction algorithm on the code word to detect and/or correct errors in the code word. The error detection and correction module 210 then extracts the data bits from the code word and outputs a corresponding data block.

[0031] In an embodiment with the code word having 288 bits, the code word includes 24 error detection and correction bits and 264 data (or payload) bits and can correct a string of eight adjacent erroneous bits. Each memory device contributes two 4-bit nibbles to each code word which are adjacent in the code word and the system can correct for the failure of any one of the 36 memory devices, thereby achieving chipkill.

[0032] Although the exemplary embodiment is described above with regard to a system where the memory devices 202 have the same width, embodiments of the invention encompass the methods described above applied with nibbles and/or memory devices in a system having different widths. Embodiments of the invention also encompass a nibble having a bit width spanning multiple memory devices and being stored into a single address line of multiple memory devices. For example, a nibble having a bit width of 8 may be stored in two 4-bit memory devices.

[0033] The components of the system 200 shown in Figure 2 are for illustrative purposes only and the invention is not limited to separate modules performing the methods of the invention. The functions of the invention may be performed by a single module or may be performed by multiple modules. For example, the functions of the data separator 208 and the data combiner may be performed by one module or the buffer 204, data separator 208, data combiner 206, and error detection and correction module 210 may be combined into a single module that performs their functions.

[0034] The memory devices 202 can be for example, but not limited to, single-in-line memory modules (SIMM), dual-in-line memory modules (DIMM) and dynamic random access memory (DRAM) modules or other types of memory devices.

[0035] The term "nibble" as used herein is not limited to any particular number of bits. A nibble is used herein to identify a number of adjacent bits where

that number can vary in accordance with the methods and systems encompassed by the invention.

[0036] Although embodiments of the invention are described above with regard to reading and writing chunks from memory, the reading and writing of multiple chunks are not necessarily separate operations. Several chunks may be read from or written to memory devices in block read and write operation. In an embodiment of the invention, the chunks corresponding to a code word are all read in a single block read or are all written in a single block write to the memory devices.

[0037] The foregoing describes the invention in terms of embodiments foreseen by the inventors for which an enabling description was available, although insubstantial modifications of the invention, not presently foreseen may nonetheless represent equivalents thereto.